

# Discourse Representation Structure Parsing



# Discourse Representation Structure Parsing

**Jiangming Liu** **Shay B. Cohen** **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
Jiangming.Liu@ed.ac.uk, scohen@inf.ed.ac.uk, mlap@inf.ed.ac.uk

## Abstract

We introduce an open-domain neural semantic parser which generates formal meaning representations in the style of Discourse Representation Theory (DRT; Kamp and Reyle 1993). We propose a method which transforms Discourse Representation Structures (DRSs) to trees and develop a structure-aware model which decomposes the decoding process into three stages: basic DRS structure prediction, condition prediction (i.e., predicates and relations), and referent prediction (i.e., variables). Experimental results on the Groningen Meaning Bank (GMB) show that our model outperforms competitive baselines by a wide margin.

## 1 Introduction

Semantic parsing is the task of mapping natural language to machine interpretable meaning representations. A variety of meaning representations have been adopted over the years ranging from functional query language (FunQL; Kate et al. 2005) to dependency-based compositional semantics (λ-DCS; Liang et al. 2011), lambda calculus (Zettlemoyer and Collins, 2005), abstract meaning representations (Banarescu et al., 2013), and minimal recursion semantics (Copestake et al., 2005).

Existing semantic parsers are for the most part data-driven using annotated examples consisting of utterances and their meaning representations (Zelle and Mooney, 1996; Wong and Mooney, 2006; Zettlemoyer and Collins, 2005). The successful application of encoder-decoder models (Sutskever et al., 2014; Bahdanau et al., 2015) to a variety of NLP tasks has provided strong impetus to treat semantic parsing as a sequence transduction problem where an utterance is mapped to a target meaning representation in string format (Dong and Lapata, 2016; Jia and Liang, 2016; Kočiskỳ et al., 2016). The fact that meaning representations do not naturally conform to a linear ordering has also prompted efforts to develop recurrent neural network architectures tailored to tree or graph-structured decoding (Dong and Lapata, 2016; Cheng et al., 2017; Yin and Neubig, 2017; Alvarez-Melis and Jaakkola, 2017; Rabinovich et al., 2017; Buys and Blunsom, 2017)

Most previous work focuses on building semantic parsers for question answering tasks, such as querying a database to retrieve an answer (Zelle and Mooney, 1996; Cheng et al., 2017), or conversing with a flight booking system (Dahl et al., 1994). As a result, parsers trained on query-based datasets work on restricted domains (e.g., restaurants, meetings; Wang et al. 2015), with limited vocabularies, exhibiting limited compositionality, and a small range of syntactic and semantic constructions. In this work, we focus on open-domain semantic parsing and develop a general-purpose system which generates formal meaning representations in the style of Discourse Representation Theory (DRT; Kamp and Reyle 1993).

DRT is a popular theory of meaning representation designed to account for a variety of linguistic phenomena, including the interpretation of pronouns and temporal expressions within and across sentences. Advantageously, it supports meaning representations for entire texts rather than isolated sentences which in turn can be translated into first-order logic. The Groningen Meaning Bank (GMB; Bos et al. 2017) provides a large collection of English texts annotated with Discourse Representation Structures (see Figure 1 for an example). GMB integrates various levels of semantic annotation (e.g., anaphora, named entities, thematic roles, rhetorical relations) into a unified formalism providing expressive meaning representations for open-domain texts.

We treat DRT parsing as a structure prediction problem. We develop a method to transform DRSs to tree-based representations which can be further linearized to bracketed string format. We examine a series of encoder-decoder models (Bahdanau et al., 2015) differing in the way tree-

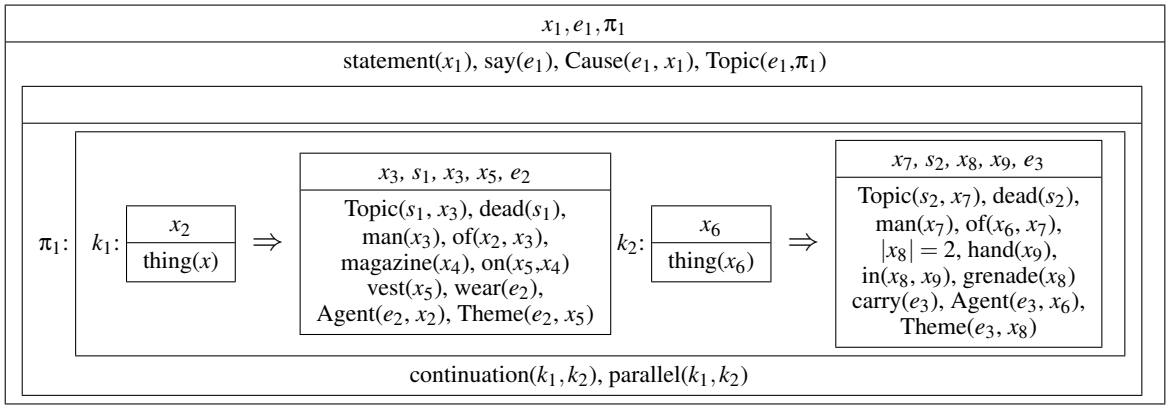


Figure 1: DRT meaning representation for the sentence *The statement says each of the dead men wore magazine vests and carried two hand grenades.*

structured logical forms are generated and show that a structure-aware decoder is paramount to open-domain semantic parsing. Our proposed model decomposes the decoding process into three stages. The first stage predicts the structure of the meaning representation omitting details such as predicates or variable names. The second stage fills in missing predicates and relations (e.g., thing, Agent) conditioning on the natural language input and the previously predicted structure. Finally, the third stage predicts variable names based on the input and the information generated so far.

Decomposing decoding into these three steps reduces the complexity of generating logical forms since the model does not have to predict deeply nested structures, their variables, and predicates all at once. Moreover, the model is able to take advantage of the GMB annotations more efficiently, e.g., examples with similar structures can be effectively used in the first stage despite being very different in their lexical make-up. Finally, a piecemeal mode of generation yields more accurate predictions; since the output of every decoding step serves as input to the next one, the model is able to refine its predictions taking progressively more global context into account. Experimental results on the GMB show that our three-stage decoder outperforms a vanilla encoder-decoder model and a related variant which takes shallow structure into account, by a wide margin.

Our contributions in this work are three-fold: an open-domain semantic parser which yields discourse representation structures; a novel end-to-end neural model equipped with a structured decoder which decomposes the parsing process into three stages; a DRS-to-tree conversion method which transforms DRSs to tree-based representations allowing for the application of structured decoders as well as sequential modeling. We release our code[1] and tree formatted version of the GMB in the hope of driving further research in open-domain semantic parsing.

## 2 Discourse Representation Theory

In this section we provide a brief overview of the representational semantic formalism used in the GMB. We refer the reader to Bos et al. (2017) and Kamp and Reyle (1993) for more details.

Discourse Representation Theory (DRT; Kamp and Reyle 1993) is a general framework for representing the meaning of sentences and discourse which can handle multiple linguistic phenomena including anaphora, presuppositions, and temporal expressions. The basic meaning-carrying units in DRT are Discourse Representation Structures (DRSs), which are recursive formal meaning structures that have a model-theoretic interpretation and can be translated into first-order logic (Kamp and Reyle, 1993). Basic DRSs consist of discourse referents (e.g., $x, y$) representing entities in the discourse and discourse conditions (e.g., man($x$), magazine($y$)) representing information about discourse referents. Following conventions in the DRT literature, we visualize DRSs in a box-like format (see Figure 1).

GMB adopts a variant of DRT that uses a neo-Davidsonian analysis of events (Kipper et al., 2008), i.e., events are first-order entities characterized by one-place predicate symbols (e.g., say($e_1$) in Figure 1). In addition, it follows Projective Discourse Representation Theory (PDRT; Venhuizen et al. 2013) an extension of DRT specifically developed to account for the interpretation of presuppositions and related projection phenomena

[1] https://github.com/EdinburghNLP/EncDecDRSparsing

(e.g., conventional implicatures). In PDRT, each basic DRS introduces a label, which can be bound by a pointer indicating the interpretation site of semantic content. To account for the rhetorical structure of texts, GMB adopts Segmented Discourse Representation Theory (SDRT; Asher and Lascarides 2003). In SDRT, discourse segments are linked with rhetorical relations reflecting different characteristics of textual coherence, such as temporal order and communicative intentions (see continuation($k_1$, $k_2$) in Figure 1).

More formally, DRSs are expressions of type $\langle exp_e \rangle$ (denoting individuals or discourse referents) and $\langle exp_t \rangle$ (i.e., truth values):

$$\langle exp_e \rangle ::= \langle ref \rangle, \quad \langle exp_t \rangle ::= \langle drs \rangle | \langle sdrs \rangle, \tag{1}$$

discourse referents $\langle ref \rangle$ are in turn classified into six categories, namely common referents ($x_n$), event referents ($e_n$), state referents ($s_n$), segment referents ($k_n$), proposition referents ($\pi_n$), and time referents ($t_n$). $\langle drs \rangle$ and $\langle sdrs \rangle$ denote basic and segmented DRSs, respectively:

$$\langle drs \rangle ::= \langle pvar \rangle : \boxed{\frac{(\langle pvar \rangle, \langle ref \rangle)^*}{(\langle pvar \rangle, \langle condition \rangle)^*}}, \tag{2}$$

$$\langle sdrs \rangle ::= \boxed{\frac{k_1 : \langle exp_t \rangle, k_2 : \langle exp_t \rangle}{coo(k_1, k_2)}} \mid \boxed{\frac{\substack{k_1:\langle exp_t \rangle \\ k_2:\langle exp_t \rangle}}{sub(k_1, k_2)}}, \tag{3}$$

Basic DRSs consist of a set of referents ($\langle ref \rangle$) and conditions ($\langle condition \rangle$), whereas segmented DRSs are *recursive* structures that combine two $\langle exp_t \rangle$ by means of coordinating (*coo*) or subordinating (*sub*) relations. DRS conditions can be basic or complex:

$$\langle condition \rangle ::= \langle basic \rangle | \langle complex \rangle, \tag{4}$$

Basic conditions express properties of discourse referents or relations between them:

$$\begin{aligned} \langle basic \rangle ::= & \langle sym_1 \rangle (\langle exp_e \rangle) \mid \langle sym_2 \rangle (\langle exp_e \rangle, \langle exp_e \rangle) \\ & \mid \langle exp_e \rangle = \langle exp_e \rangle \mid \langle exp_e \rangle = \langle num \rangle \\ & \mid timex(\langle exp_e \rangle, \langle sym_0 \rangle) \\ & \mid named(\langle exp_e \rangle, \langle sym_0 \rangle, class). \end{aligned} \tag{5}$$

where $\langle sym_n \rangle$ denotes $n$-place predicates, $\langle num \rangle$ denotes cardinal numbers, *timex* expresses temporal information (e.g., timex($x_7$, 2005) denotes the year 2005), and *class* refers to named entity classes (e.g., location).

Complex conditions are unary or binary. Unary conditions have one DRS as argument and represent negation ($\neg$) and modal operators expressing necessity ($\Box$) and possibility ($\Diamond$). Condition $\langle ref \rangle : \langle exp_t \rangle$ represents verbs with propositional content (e.g., factive verbs). Binary conditions are conditional statements ($\rightarrow$) and questions.

$$\langle complex \rangle ::= \langle unary \rangle \mid \langle binary \rangle, \tag{6}$$
$$\langle unary \rangle ::= \neg \langle exp_t \rangle \mid \Box \langle exp_t \rangle | \Diamond \langle exp_t \rangle | \langle ref \rangle : \langle exp_t \rangle$$
$$\langle binary \rangle ::= \langle exp_t \rangle \rightarrow \langle exp_t \rangle | \langle exp_t \rangle \vee \langle exp_t \rangle | \langle exp_t \rangle ? \langle exp_t \rangle$$

| sections | # doc | # sent | # token | avg |
|---|---|---|---|---|
| 00-99 | 10,000 | 62,010 | 1,354,149 | 21.84 |
| 20-99 | 7,970 | 49,411 | 1,078,953 | 21.83 |
| 10-19 | 1,038 | 6,483 | 142,344 | 21.95 |
| 00-09 | 992 | 6,116 | 132,852 | 21.72 |

Table 1: Statistics on the GMB (avg denotes the average number of tokens per sentence).

## 3 The Groningen Meaning Bank Corpus

**Corpus Creation** DRSs in GMB were obtained from Boxer (Bos, 2008, 2015), and then refined using expert linguists and crowdsourcing methods. Boxer constructs DRSs based on a pipeline of tools involving POS-tagging, named entity recognition, and parsing. Specifically, it relies on the syntactic analysis of the C&C parser (Clark and Curran, 2007), a general-purpose parser using the framework of Combinatory Categorial Grammar (CCG; Steedman 2001). DRSs are obtained from CCG parses, with semantic composition being guided by the CCG syntactic derivation.

Documents in the GMB were collected from a variety of sources including *Voice of America* (a newspaper published by the US Federal Government), the Open American National Corpus, Aesop's fables, humorous stories and jokes, and country descriptions from the *CIA World Factbook*. The dataset consists of 10,000 documents each annotated with a DRS. Various statistics on the GMB are shown in Table 1. Bos et al. (2017) recommend sections 20–99 for training, 10–19 for tuning, and 00–09 for testing.

**DRS-to-Tree Conversion** As mentioned earlier, DRSs in the GMB are displayed in a box-like format which is intuitive and easy to read but not particularly amenable to structure modeling. In this section we discuss how DRSs were post-processed and simplified into a tree-based format, which served as input to our models.

The GMB provides DRS annotations per-document. Our initial efforts have focused on sentence-level DRS parsing which is undoubtedly

a necessary first step for more global semantic representations. It is relatively, straightforward to obtain sentence-level DRSs from document-level annotations since referents and conditions are indexed to tokens. We match each sentence in a document with the DRS whose content bears the same indices as the tokens occurring in the sentence. This matching process yields 52,268 sentences for training (sections 20–99), 5,172 sentences for development (sections 10–19), (development), and 5,440 sentences for testing (sections 00–09).

In order to simplify the representation, we omit referents in the top part of the DRS (e.g., $x_1$, $e_1$ and $\pi_1$ in Figure 1) but preserve them in conditions without any information loss. Also we ignore pointers to DRSs since this information is implicitly captured through the typing and co-indexing of referents. Definition (1) is simplified to:

$$\langle drs \rangle ::= \text{DRS}(\langle condition \rangle^*), \tag{7}$$

where DRS() denotes a basic DRS. We also modify discourse referents to SDRSs (e.g., $k_1$, $k_2$ in Figure 1) which we regard as elements bearing scope over expressions $\langle exp_t \rangle$ and add a 2-place predicate $\langle sym_2 \rangle$ to describe the discourse relation between them. So, definition (3) becomes:

$$\begin{aligned} \langle sdrs \rangle ::= &\text{SDRS}((\langle ref \rangle(\langle exp_t \rangle))^* \\ &(\langle sym_2 \rangle(\langle ref \rangle, \langle ref \rangle))^*), \end{aligned} \tag{8}$$

where SDRS() denotes a segmented DRS, and $\langle ref \rangle$ are segment referents.

We treat cardinal numbers $\langle num \rangle$ and $\langle sym_0 \rangle$ in relation *timex* as constants. We introduce the binary predicate "card" to represent cardinality (e.g., $|x_8| = 2$ is $\text{card}(x_8, \text{NUM})$). We also simplify $\langle exp_e \rangle = \langle exp_e \rangle$ to $\text{eq}(\langle exp_e \rangle, \langle exp_e \rangle)$ using the binary relation "eq" (e.g., $x_1 = x_2$ becomes $\text{eq}(x_1, x_2)$). Moreover, we ignore *class* in *named* and transform $named(\langle exp_e \rangle, \langle sym_0 \rangle, class)$ into $\langle sym_1 \rangle(\langle exp_e \rangle)$ (e.g., $\text{named}(x_2, \text{mongolia}, geo)$ becomes $\text{mongolia}(x_2)$). Consequently, basic conditions (see definition (5)) are simplified to:

$$\langle basic \rangle ::= \langle sym_1 \rangle(\langle exp_e \rangle) | \langle sym_2 \rangle(\langle exp_e \rangle, \langle exp_e \rangle) \tag{9}$$

Analogously, we treat *unary* and *binary* conditions as scoped functions, and definition (6) becomes:

$$\begin{aligned} \langle unary \rangle ::= &\quad \neg \mid \Box \mid \Diamond \mid \langle ref \rangle(\langle exp_t \rangle) \\ \langle binary \rangle ::= &\quad \rightarrow \mid \vee \mid ?(\langle exp_t \rangle, \langle exp_t \rangle), \end{aligned} \tag{10}$$

Following the transformations described above, the DRS in Figure 1 is converted into the tree in Figure 2, which can be subsequently linearized into a PTB-style bracketed sequence. It is important to note that the conversion does not diminish the complexity of DRSs. The average tree width in the training set is 10.39 and tree depth is 4.64.

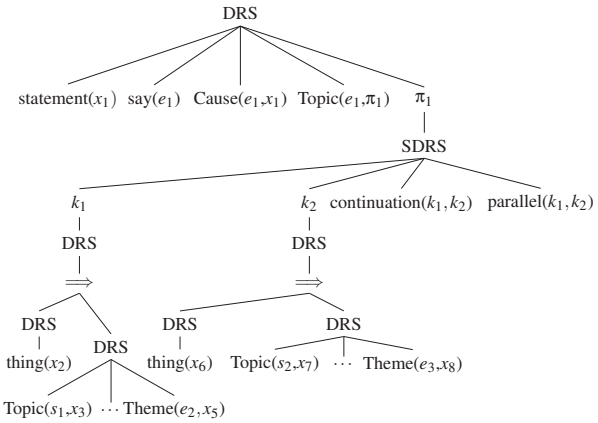


Figure 2: Tree-based representation (top) of the DRS in Figure 1 and its linearization (bottom).

## 4 Semantic Parsing Models

We present below three encoder-decoder models which are increasingly aware of the structure of the DRT meaning representations. The models take as input a natural language sentence $X$ represented as $w_1, w_2, \ldots, w_n$, and generate a sequence $Y = (y_1, y_2, ..., y_m)$, which is a linearized tree (see Figure 2 bottom), where $n$ is the length of the sentence, and $m$ the length of the generated DRS sequence. We aim to estimate $p(Y|X)$, the conditional probability of the semantic parse tree $Y$ given natural language input $X$:

$$p(Y|X) = \prod_j p(y_j | Y_1^{j-1}, X_1^n)$$

### 4.1 Encoder

An encoder is used to represent the natural language input $X$ into vector representations. Each token in a sentence is represented by a vector $x_k$ which is the concatenation of randomly initialized embeddings $e_{w_i}$, pre-trained word embeddings $\bar{e}_{w_i}$, and lemma embeddings $e_{l_i}$: $x_k = \tanh([e_{w_i}; \bar{e}_{w_i}; e_{l_i}] * W_1 + b_1)$, where $W_1 \in \mathbb{R}^{\mathcal{D}}$ and $\mathcal{D}$ is a shorthand for $(d_w + d_p + d_l) \times d_{input}$ (subscripts $w$, $p$, and $l$ denote the dimensions of word embeddings, pre-trained embeddings, and lemma embeddings, respectively); $b_1 \in \mathbb{R}^{d_{input}}$ and the symbol ; denotes concatenation. Embeddings $e_{w_i}$

and $e_{l_i}$ are randomly initialized and tuned during training, while $\bar{e}_{w_i}$ are fixed.

We use a bidirectional recurrent neural network with long short-term memory units (bi-LSTM; Hochreiter and Schmidhuber 1997) to encode natural language sentences:

$$[h_{e_1} : h_{e_n}] = \text{bi-LSTM}(x_1 : x_n),$$

where $h_{e_i}$ denotes the hidden representation of the encoder, and $x_i$ refers to the input representation of the $i$th token in the sentence. Table 2 summarizes the notation used throughout this paper.

### 4.2 Sequence Decoder

We employ a sequential decoder (Bahdanau et al., 2015) as our baseline model with the architecture shown in Figure 3(a). Our decoder is a (forward) LSTM, which is conditionally initialized with the hidden state of the encoder, i.e., we set $h_{d_0} = h_{e_n}$ and $c_{d_0} = c_{e_n}$, where $c$ is a memory cell:

$$h_{d_j} = \text{LSTM}(e_{y_{j-1}}),$$

where $h_{d_j}$ denotes the hidden representation of $y_j$, $e_{y_j}$ are randomly initialized embeddings tuned during training, and $y_0$ denotes the start of sequence.

The decoder uses the contextual representation of the encoder together with the embedding of the previously predicted token to output the next token from the vocabulary $V$:

$$s_j = [h_{ct_j}; e_{y_{j-1}}] * W_2 + b_2,$$

where $W_2 \in \mathbb{R}^{(d_{enc}+d_y)\times|V|}$, $b_2 \in \mathbb{R}^{|V|}$, $d_{enc}$ and $d_y$ are the dimensions of the encoder hidden unit and output representation, respectively, and $h_{ct_j}$ is obtained using an attention mechanism:

$$h_{ct_j} = \sum_{i=1}^{n} \beta_{ji} h_{e_i},$$

where the weight $\beta_{ji}$ is computed by:

$$\beta_{ji} = \frac{e^{f(h_{d_j}, h_{e_i})}}{\sum_k e^{f(h_{d_j}, h_{e_k})}},$$

and $f$ is the dot-product function. We obtain the probability distribution over the output tokens as:

$$p_j = p(y_j | Y_1^{j-1}, X_1^n) = \text{SoftMax}(s_j)$$

| Symbol | Description |
|---|---|
| $X; Y$ | sequence of words; outputs |
| $w_i; y_i$ | the $i$th word; output |
| $X_i^j; Y_i^j$ | word; output sequence from position $i$ to $j$ |
| $e_{w_i}; e_{y_i}$ | random embedding of word $w_i$; of output $y_i$ |
| $\bar{e}_{w_i}$ | fixed pretrained embedding of word $w_i$ |
| $e_{l_i}$ | random embedding for lemma $l_i$ |
| $d_w$ | dimension of random word embedding |
| $d_p$ | dimension of pretrained word embedding |
| $d_l$ | the dimension of random lemma embedding |
| $d_{input}$ | input dimension of encoder |
| $d_{enc}; d_{dec}$ | hidden dimension of encoder; decoder |
| $W_i$ | matrix of model parameters |
| $b_i$ | vector of model parameters |
| $x_i$ | representation of $i$th token |
| $h_{e_i}$ | hidden representation of $i$th token |
| $c_{e_i}$ | memory cell of $i$th token in encoder |
| $h_{d_i}$ | hidden representation of $i$th token in decoder |
| $c_{d_i}$ | memory cell of $i$th token in decoder |
| $s_j$ | score vector of $j$th output in decoder |
| $h_{ct_j}$ | context representation of $j$th output |
| $\beta_j^i$ | alignment from $j$th output to $i$th token |
| $o_j^i$ | copy score of $j$th output from $i$th token |
| $\hat{}$ | indicates tree structure (e.g. $\hat{Y}, \hat{y}_i, \hat{s}_j$) |
| $\bar{}$ | indicates DRS conditions (e.g. $\bar{Y}, \bar{y}_i, \bar{s}_j$) |
| $\dot{}$ | indicates referents (e.g. $\dot{Y}, \dot{y}_i, \dot{s}_j$) |

Table 2: Notation used throughout this paper.

### 4.3 Shallow Structure Decoder

The baseline decoder treats all conditions in a DRS uniformly and has no means of distinguishing between conditions corresponding to tokens in a sentence (e.g., the predicate say($e_1$) refers to the verb *said*) and semantic relations (e.g., Cause($e_1, x_1$)). Our second decoder attempts to take this into account by distinguishing conditions which are local and correspond to words in a sentence from items which are more global and express semantic content (see Figure 3(b)). Specifically, we model sentence specific conditions using a copying mechanism, and all other conditions $\mathcal{G}$ which do not correspond to sentential tokens (e.g., thematic roles, rhetorical relations) with an insertion mechanism.

Each token in a sentence is assigned a copying score $o_{ji}$:

$$o_{ji} = h_{d_j}^\top W_3 h_{e_i},$$

where subscript $ji$ denotes the $i$th token at $j$th time step, and $W_3 \in \mathbb{R}^{d_{dec} \times d_{enc}}$. All other conditions $\mathcal{G}$ are assigned an insertion score:

$$s_j = [h_{ct_j}; e_{y_{j-1}}] * W_4 + b_4,$$

where $W_4 \in \mathbb{R}^{(d_{enc}+d_y)\times|\mathcal{G}|}$, $b_4 \in \mathbb{R}^{|\mathcal{G}|}$, and $h_{ct_j}$ are the same with the baseline decoder. We obtain the probability distribution over output tokens as:

$$p_j = p(y_j | Y_1^{j-1}, X_1^n) = \text{SoftMax}([o_j; s_j])$$

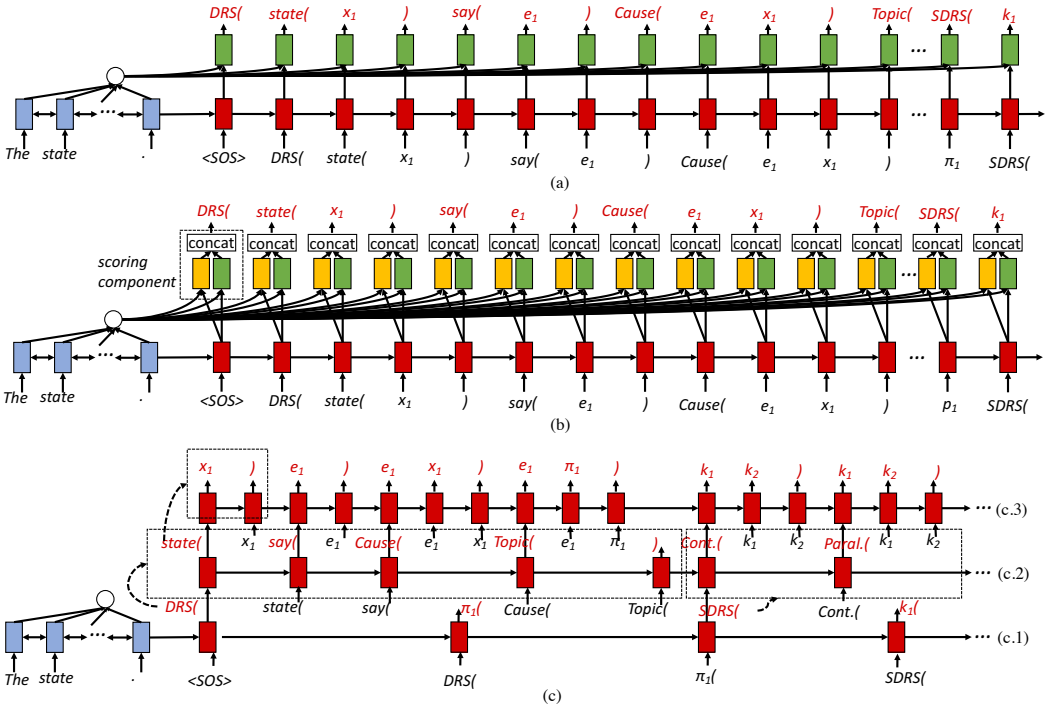


Figure 3: (a) baseline model; (b) shallow structure model; (c) deep structure model (scoring components are not displayed): (c.1) predicts DRS structure, (c.2) predicts conditions, and (c.3) predicts referents. Blue boxes are encoder hidden units, red boxes are decoder LSTM hidden units, green and yellow boxes represent copy and insertion scores, respectively.

### 4.4 Deep Structure Decoder

As explained previously, our structure prediction problem is rather challenging: the length of a bracketed DRS is nearly five times longer than its corresponding sentence. As shown in Figure 1, a bracketed DRS, $y_1, y_2, ..., y_n$ consists of three parts: internal structure $\hat{Y} = \hat{y}_1, \hat{y}_2, ...\hat{y}_t$ (e.g., DRS( $\pi_1$( SDRS($k_1$(DRS(→(DRS( )DRS( ))) $k_2$( DRS(→( DRS( ) DRS ( ) ) ) ) ) ) )), conditions $\bar{Y} = \bar{y}_1, \bar{y}_2, ..., \bar{y}_r$ (e.g., statement, say, Topic), and referents $\dot{Y} = \dot{y}_1, \dot{y}_2, ..., \dot{y}_v$ (e.g., $x_1$, $e_1$, $\pi_1$), where $t + r * 2 + v = n$.[2]

Our third decoder (see Figure 3(c)) first predicts the structural make-up of the DRS, then the conditions, and finally their referents in an end-to-end framework. The probability distribution of structured output $Y$ given natural language input $X$ is rewritten as:

$$\begin{aligned} p(Y|X) = \; & p(\hat{Y}, \bar{Y}, \dot{Y}|X) \\ = \; & \prod_j p(\hat{y}_j|\hat{Y}_1^{j-1}, X) \\ & \times \prod_j p(\bar{y}_j|\bar{Y}_1^{j-1}, \hat{Y}_1^{j'}, X) \\ & \times \prod_j p(\dot{y}_j|\dot{Y}_1^{j-1}, \bar{Y}_1^{j'}, \hat{Y}_1^{j''}, X) \end{aligned} \quad (11)$$

where $\hat{Y}_1^{j-1}$, $\bar{Y}_1^{j-1}$, and $\dot{Y}_1^{j-1}$ denote the tree structure, conditions, and referents predicted so far. $\hat{Y}_1^{j'}$ denotes the structure predicted *before* conditions $\bar{y}_j$; $\hat{Y}_1^{j''}$ and $\bar{Y}_1^{j'}$ are the structures and conditions predicted *before* referents $\dot{y}_j$. We next discuss how each decoder is modeled.

**Structure Prediction** To model basic DRS structure we apply the shallow decoder discussed in Section 4.3 and also shown in Figure 3(c.1). Tokens in such structures correspond to parent nodes in a tree; in other words, they are all inserted from $\mathcal{G}$, and subsequently predicted tokens are only scored with the insert score, i.e., $\hat{s}_i = s_i$. The hidden units of the decoder are:

$$\hat{h}_{d_j} = \text{LSTM}(e_{\hat{y}_{j-1}}),$$

And the probabilistic distribution over structure denoting tokens is:

$$p(y_j|Y_1^{j-1}, X) = \text{SoftMax}(\hat{s}_j)$$

**Condition Prediction** DRS conditions are generated by taking previously predicted structures into account, e.g., when "DRS(" or "SDRS(" are predicted, their conditions will be generated next. By mapping $j$ to $(k, m_k)$, the sequence of conditions can be rewritten as $\bar{y}_1, \ldots, \bar{y}_j, \ldots, \bar{y}_r = \bar{y}_{(1,1)}, \bar{y}_{(1,2)}, \ldots, \bar{y}_{(k,m_k)}, \ldots$, where $\bar{y}_{(k,m_k)}$ is $m_k$th

[2]Each condition has one and only one right bracket.

condition of structure token $\hat{y}_k$. The corresponding hidden units $\hat{h}_{d_k}$ act as conditional input to the decoder. Structure denoting tokens (e.g., "DRS(" or "SDRS(") are fed into the decoder one by one to generate the corresponding conditions as:

$$e_{\bar{y}_{(k,0)}} = \hat{h}_{d_k} * W_5 + b_5,$$

where $W_5 \in \mathbb{R}^{d_{dec} \times d_y}$ and $b_5 \in \mathbb{R}^{d_y}$. The hidden unit of the conditions decoder is computed as:

$$\bar{h}_{d_j} = \bar{h}_{d_{(k,m_k)}} = \text{LSTM}(e_{\bar{y}_{(k,m_k-1)}}),$$

Given hidden unit $\bar{h}_{d_j}$, we obtain the copy score $\bar{o}_j$ and insert score $\bar{s}_j$. The probabilistic distribution over conditions is:

$$p(\bar{y}_j|\bar{Y}_1^{j-1}, \hat{Y}_1^{j'}, X) = \text{SoftMax}([\bar{o}_j; \bar{s}_j])$$

**Referent Prediction** Referents are generated based on the structure and conditions of the DRS. Each condition has at least one referent. Similar to condition prediction, the sequence of referents can be rewritten as $\dot{y}_1, \ldots, \dot{y}_j, \ldots, \dot{y}_v = \dot{y}_{(1,1)}, \dot{y}_{(1,2)}, \ldots, \dot{y}_{(k,m_k)}, \ldots$ The hidden units of the conditions decoder are fed into the referent decoder $e_{\dot{y}_{(k,0)}} = \bar{h}_{d_k} * W_6 + b_6$, where $W_6 \in \mathbb{R}^{d_{dec} \times d_y}$, $b_6 \in \mathbb{R}^{d_y}$. The hidden unit of the referent decoder is computed as:

$$\dot{h}_{d_j} = \dot{h}_{d_{(k,m_k)}} = \text{LSTM}(e_{\dot{y}_{(k,m_k-1)}}),$$

All referents are inserted from $\mathcal{G}$, given hidden unit $\dot{h}_{d_j}$ (we only obtain the insert score $\dot{s}_j$). The probabilistic distribution over predicates is:

$$p(\dot{y}_j|\dot{Y}_1^{j-1}, \bar{Y}_1^{j'}, \hat{Y}_1^{j''}, X) = \text{SoftMax}(\dot{s}_j).$$

Note that a single LSTM is adopted for structure, condition and referent prediction. The mathematic symbols are summarized in Table 2.

### 4.5 Training

The models are trained to minimize a cross-entropy loss objective with $\ell_2$ regularization:

$$L(\theta) = -\sum_j \log p_j + \frac{\lambda}{2}||\theta||^2,$$

where $\theta$ is the set of parameters, and $\lambda$ is a regularization hyper-parameter ($\lambda = 10^{-6}$). We used stochastic gradient descent with Adam (Kingma and Ba, 2014) to adjust the learning rate.

## 5 Experimental Setup

**Settings** Our experiments were carried out on the GMB following the tree conversion process discussed in Section 3. We adopted the training, development, and testing partitions recommended in Bos et al. (2017). We compared the three models introduced in Section 4, namely the baseline sequence decoder, the shallow structured decoder and the deep structure decoder. We used the same empirical hyper-parameters for all three models. The dimensions of word and lemma embeddings were 64 and 32, respectively. The dimensions of hidden vectors were 256 for the encoder and 128 for the decoder. The encoder used two hidden layers, whereas the decoder only one. The dropout rate was 0.1. Pre-trained word embeddings (100 dimensions) were generated with Word2Vec trained on the AFP portion of the English Gigaword corpus.[3]

**Evaluation** Due to the complex nature of our structured prediction task, we cannot expect model output to exactly match the gold standard. For instance, the numbering of the referents may be different, but nevertheless valid, or the order of the children of a tree node (e.g., "DRS(india($x_1$) say($e_1$))" and "DRS(say($e_1$) india($x_1$))" are the same). We thus use $F_1$ instead of exact match accuracy. Specifically, we report D-match[4] a metric designed to evaluate scoped meaning representations and released as part of the distribution of the Parallel Meaning Bank corpus (Abzianidze et al., 2017). D-match is based on Smatch[5], a metric used to evaluate AMR graphs (Cai and Knight, 2013); it calculates $F_1$ on discourse representation graphs (DRGs), i.e., triples of nodes, arcs, and their referents, applying multiple restarts to obtain a good referent (node) mapping between graphs.

We converted DRSs (predicted and goldstandard) into DRGs following the top-down procedure described in Algorithm 1.[6] IsCondition returns *true* if the child is a condition (e.g., india($x_1$)), where three arcs are created, one is connected to a parent node and the other two are connected to arg1 and arg2, respectively (lines 7–12). IsQuantifier returns *true* if the child is a quantifier (e.g., $\pi_1$, $\neg$ and $\Box$) and three arcs are created; one is connected to the parent node, one to the referent that is created if and only

[3] The models are trained on a single GPU without batches.
[4] https://github.com/RikVN/D-match
[5] https://github.com/snowblink14/smatch
[6] We refer the interested reader to the supplementary material for more details.

**Algorithm 1** DRS to DRG Conversion

**Input**: $T$, tree-like DRS
**Output**: $G$, a set of edges

1: $n_b \leftarrow 0; n_c \leftarrow 0; G \leftarrow \emptyset$
2: $stack \leftarrow []; R \leftarrow \emptyset$
3: **procedure** TRAVELDRS(parent)
4: $stack.append(b_{n_b}); n_b \leftarrow n_b + 1$
5: $node_p \leftarrow stack.top$
6: **for** *child* **in** *parent* **do**
7: **if** ISCONDITION(*child*) **then**
8: $G \leftarrow G \cup \{node_p \xrightarrow{child.rel} c_{n_c}\}$
9: $G \leftarrow G \cup \{c_{n_c} \xrightarrow{\text{arg1}} child.arg1\}$
10: $G \leftarrow G \cup \{c_{n_c} \xrightarrow{\text{arg2}} child.arg2\}$
11: $n_c \leftarrow n_c + 1$
12: ADDREFERENT($node_p, child$)
13: **else if** ISQUANTIFIER(*child*) **then**
14: $G \leftarrow G \cup \{node_p \xrightarrow{child.class} c_{n_c}\}$
15: $G \leftarrow G \cup \{c_{n_c} \xrightarrow{\text{arg1}} child.arg1\}$
16: $G \leftarrow G \cup \{c_{n_c} \xrightarrow{\text{arg1}} b_{n_b+1}\}$
17: $n_c \leftarrow n_c + 1$
18: **if** ISPROPSEG(*child*) **then**
19: ADDREFERENT($node_p, child$)
20: **end if**
21: TRAVELDRS($child.nextDRS$)
22: **end if**
23: **end for**
24: $stack.pop()$
25: **end procedure**
26: **procedure** ADDREFERENT($node_p, child$)
27: **if** $child.arg1$ **not in** $R$ **then**
28: $G \leftarrow G \cup \{node_p \xrightarrow{\text{ref}} child.arg1\}$
29: $R \leftarrow R \cup child.arg1$
30: **end if**
31: **if** $child.arg2$ **not in** $R$ **then**
32: $G \leftarrow G \cup \{node_p \xrightarrow{\text{ref}} child.arg2\}$
33: $R \leftarrow R \cup child.arg2$
34: **end if**
35: **end procedure**
36: TRAVELDRS($T$)
37: return $G$

if the child is a proposition or segment (e.g., $\pi_1$ and $k_1$), and one is connected to the next DRS or SDRS nodes (lines 13–20). The algorithm will recursively travel all DRS or SDRS nodes (line 21). Furthermore, arcs are introduced to connect DRS or SDRS nodes to the referents that first appear in a condition (lines 26–35).

When comparing two DRGs, we calculate the $F_1$ over their arcs. For example consider the two DRGs (a) and (b) shown in Figure 4. Let $\{b_0 : b_0, x_1 : x_2, x_2 : x_3, c_0 : c_0, c_1 : c_2, c_2 : c_3\}$ denote the node alignment between them. The number of matching arcs is eight, the number of arcs in the gold DRG is nine, and the number of arcs in the predicted DRG is 12. So recall is 8/9, precision is 8/12, and $F_1$ is 76.19.

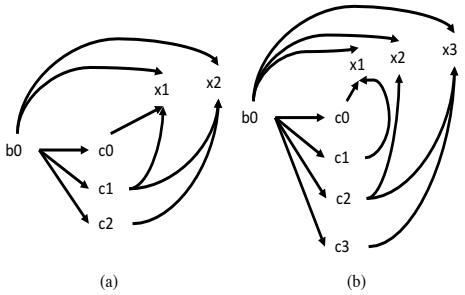


Figure 4: (a) is the gold DRS and (b) is the predicted DRS (condition names are not shown).

## 6 Results

Table 3 compares our three models on the development set. As can be seen, the shallow structured decoder performs better than the baseline decoder, and the proposed deep structure decoder outperforms both of them. Ablation experiments show that without pre-trained word embeddings or word lemma embeddings, the model generally performs worse. Compared to lemma embeddings, pre-trained word embeddings contribute more.

Table 4 shows our results on the test set. To assess the degree to which the various decoders contribute to DRS parsing, we report results when predicting the full DRS structure (second block), when ignoring referents (third block), and when ignoring both referents and conditions (fourth block). Overall, we observe that the shallow structure model improves precision over the baseline with a slight loss in recall, while the deep structure model performs best by a large margin. When referents are not taken into account (compare the second and third blocks in Table 4), performance improves across the board. When conditions are additionally omitted, we observe further performance gains. This is hardly surprising, since errors propagate from one stage to the next when predicting full DRS structures. Further analysis revealed that the parser performs slightly better on (copy) conditions which correspond to natural language tokens compared to (insert) conditions (e.g., Topic, Agent) which are generated from global semantic content (83.22 vs 80.63 $F_1$). The parser is also better on sentences which do not represent SDRSs (79.12 vs 68.36 $F_1$) which is expected given that they usually correspond to more elaborate structures. We also found that rhetorical relations (linking segments) are predicted fairly accurately, especially if they are frequently attested (e.g., Continuation, Parallel), while the parser has difficulty with relations denoting contrast.

| Model | P (%) | R (%) | $F_1$ (%) |
|---|---|---|---|
| baseline | 51.35 | 63.85 | 56.92 |
| shallow | 67.88 | 63.53 | 65.63 |
| deep | 79.01 | 75.65 | 77.29 |
| deep (–pre) | 78.47 | 73.43 | 75.87 |
| deep (–pre & lem) | 78.21 | 72.82 | 75.42 |

Table 3: GMB development set.

| Model | DRG | | | DRG w/o refs | | | DRG w/o refs & conds | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | $F_1$ | P | R | $F_1$ | P | R | $F_1$ |
| baseline | 52.21 | 64.46 | 57.69 | 47.20 | 58.93 | 52.42 | 52.89 | 71.80 | 60.91 |
| shallow | 66.61 | 63.92 | 65.24 | 66.05 | 62.93 | 64.45 | 83.30 | 62.91 | 71.68 |
| deep | 79.27 | 75.88 | 77.54 | 82.87 | 79.40 | 81.10 | 93.91 | 88.51 | 91.13 |

Table 4: GMB test set.

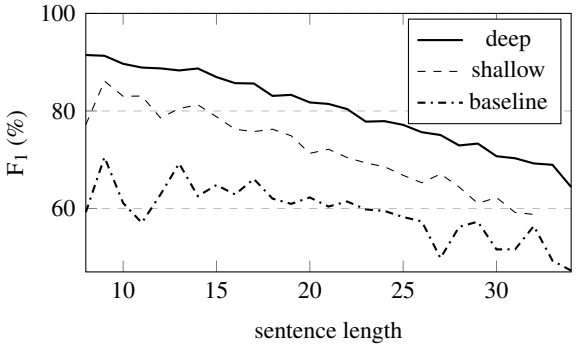


Figure 5: $F_1$ score as a function of sentence length.

Figure 5 shows $F_1$ performance for the three parsers on sentences of different length. We observe a similar trend for all models: as sentence length increases, model performance decreases. The baseline and shallow models do not perform well on short sentences which despite containing fewer words, can still represent complex meaning which is challenging to capture sequentially. On the other hand, the performance of the deep model is relatively stable. LSTMs in this case function relatively well, as they are faced with the easier task of predicting meaning in different stages (starting with a tree skeleton which is progressively refined). We provide examples of model output in the supplementary material.

## 7 Related Work

**Tree-structured Decoding** A few recent approaches develop structured decoders which make use of the syntax of meaning representations. Dong and Lapata (2016) and Alvarez-Melis and Jaakkola (2017) generate trees in a top-down fashion, while in other work (Xiao et al., 2016; Krishnamurthy et al., 2017) the decoder generates from a grammar that guarantees that predicted logical forms are well-typed. In a similar vein, Yin and Neubig (2017) generate abstract syntax trees (ASTs) based on the application of production rules defined by the grammar. Rabinovich et al. (2017) introduce a modular decoder whose various components are dynamically composed according to the generated tree structure. In comparison, our model does not use grammar information explicitly. We first decode the structure of the DRS, and then fill in details pertaining to its semantic content. Our model is not strictly speaking top-down, we generate partial trees sequentially, and then expand non-terminal nodes, ensuring that when we generate the children of a node, we have already obtained the structure of the entire tree.

**Wide-coverage Semantic Parsing** Our model is trained on the GMB (Bos et al., 2017), a richly annotated resource in the style of DRT which provides a unique opportunity for bootstrapping wide-coverage semantic parsers. Boxer (Bos, 2008) was a precursor to the GMB, the first semantic parser of this kind, which deterministically maps CCG derivations onto formal meaning representations. Le and Zuidema (2012) were the first to train a semantic parser on an early release of the GMB (2,000 documents; Basile et al. 2012), however, they abandon lambda calculus in favor of a graph based representation. The latter is closely related to AMR, a general-purpose meaning representation language for broad-coverage text. In AMR the meaning of a sentence is represented as a rooted, directed, edge-labeled and leaf-labeled graph. AMRs do not resemble classical meaning representations and do not have a model-theoretic interpretation. However, see Bos (2016) and Artzi et al. (2015) for translations to first-order logic.

## 8 Conclusions

We introduced a new end-to-end model for open-domain semantic parsing. Experimental results on the GMB show that our decoder is able to recover discourse representation structures to a good degree (77.54 $F_1$), albeit with some simplifications. In the future, we plan to model document-level representations which are more in line with DRT and the GMB annotations.

**Acknowledgments** We thank the anonymous reviewers for their feedback and Johan Bos for answering several questions relating to the GMB. We gratefully acknowledge the support of the European Research Council (Lapata, Liu; award number 681760) and the EU H2020 project SUMMA (Cohen, Liu; grant agreement 688139).

## References


Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247, Valencia, Spain.

David Alvarez-Melis and Tommi S. Jaakkola. 2017. Tree-structured decoding with doubly-recurrent neural networks. In *Proceedings of the 5th International Conference on Learning Representation (ICLR)*, Toulon, France.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal.

Nicholas Asher and Alex Lascarides. 2003. *Logics of conversation*. Cambridge University Press.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Diego, California.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pages 277–286.

Johan Bos. 2015. Open-domain semantic parsing with Boxer. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 301–304. Linköping University Electronic Press, Sweden.

Johan Bos. 2016. Expressive power of abstract meaning representations. *Computational Linguistics*, 42(3):527–535.

Johan Bos, Valerio Basile, Kilian Evang, Noortje Venhuizen, and Johannes Bjerva. 2017. The groningen meaning bank. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, volume 2, pages 463–496. Springer.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1215–1226, Vancouver, Canada.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 44–55, Vancouver, Canada.

Stephen Clark and James Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Ann Copestake, Dan Flickinger, Carl Pollar, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 2–3(3):281–332.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, Christine Pao David Pallett, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: the atis-3 corpus. In *Proceedings of the workshop on ARPA Human Language Technology*, pages 43–48, Plainsboro, New Jersey.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany.

Hans Kamp and Uwe Reyle. 1993. From discourse to logic; an introduction to modeltheoretic semantics of natural language, formal logic and DRT.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 1062–1068, Pittsburgh, PA.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, Banff, Canada.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation*, 42(1):21–40.

Tomáš Kočiskỳ, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1078–1087, Austin, Texas.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark.

Phong Le and Willem Zuidema. 2012. Learning compositional semantics for open domain semantic parsing. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 1535–1552, Mumbai, India.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon.

Ella Rabinovich, Noam Ordan, and Shuly Wintner. 2017. Found in translation: Reconstructing phylogenetic language trees from translations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 530–540, Vancouver, Canada.

Mark Steedman. 2001. *The Syntactic Process*. The MIT Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Noortje J. Venhuizen, Johan Bos, and Harm Brouwer. 2013. Parsimonious semantic representations with projection pointers. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 252–263, Potsdam, Germany.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, Beijing, China.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 439–446, New York City, USA.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1341–1350, Berlin, Germany.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1050–1055, Portland, Oregon.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *PProceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pages 658–666, Edinburgh, Scotland.